NODE - dependencias

Existen dos tipos de dependencias, las de producción y las de desarrollo.

Las dependencias de producción son los paquetes que el proyecto necesita para funcionar.

Las dependencias de desarrollo son los paquetes que utilizamos nosotros para desarrollar pero que el proyecto no necesita para su funcionamiento.

NODE - dependencias

En versiones anteriores de node había que diferenciarlas utilizando los comandos

npm install <package> --save Para guardarla como dependencia de producción.

npm install <package> --save-dev Para guardarla como dependencia de desarrollo.

En versiones actuales sólo necesitamos el comando --save-dev o -D para indicar que es dependencia de desarrollo, mientras que las dependencias de producción no necesitan ninguna “flag”

NODE - express

El módulo http es bastante complejo de utilizar cuando intentamos añadir funcionalidad real a un servidor web, para ello se creó el paquete express que simplifica muchísimo toda la lógica de un servidor web.

Para instalarlo utilizaremos el comando npm install express, express se necesita para el funcionamiento de la aplicación, por lo que no irá como dependencia de desarrollo.

NODE - express

Para crear el servidor lo haremos de una forma muy similar a como lo hacíamos con el módulo http.

```
const express = require('express');
const app = express();

// Iniciar el servidor
app.listen(port, () => {
  console.log(`El servidor está funcionando en el puerto ${port}`);
});
```

NODE - express

Para el caso en el que tenemos que reaccionar a una conexión existen varias formas de hacerlo, empecemos por la más sencilla.

A través del objeto app podemos acceder a su método get para iniciar el evento de escucha hacia la ruta raíz, como segundo parámetro tenemos una función de callback que se ejecutará cuando se reciba una solicitud a la ruta especificada.

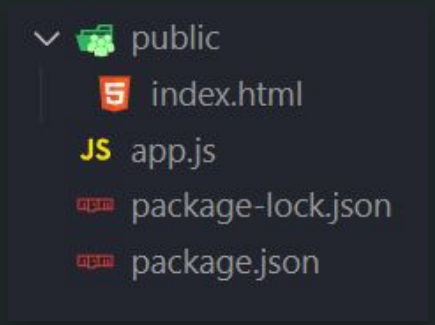
```
app.get('/', (req, res) => {
  res.send('Esta es la página principal');
});
```

```
app.get('/about', (req, res) => {
  res.send('Esta es la página about');
});
```

NODE - express - sendFile

Con el método send de la respuesta podemos enviar mensajes, pero si queremos enviar archivos html, necesitaremos usar el método sendFile()

Éstos archivos html los colocaremos en la carpeta /public.



NODE - express - sendFile

```
app.get('/', (req, res) => {
  res.sendFile('public/index.html');
});
```

Si ejecutamos este código, node nos lanzará un error porque las ruta que espera debe ser absoluta, para ello tenemos en node una variable global llamada __dirname que apunta siempre a la ruta absoluta del archivo que usa esa variable.

```
console.log(__dirname);
```

```
doria@Dorian MINGW64 /f/Users/Dorian/Desktop/node-desde-0
$ node app.js
F:\Users\Dorian\Desktop\node-desde-0
El servidor está funcionando en el puerto 8000
```

NODE - express - sendFile

Para evitar este error utilizaremos la ruta añadiendo la variable __dirname al principio, lo podemos concatenar o utilizar un template string, ambas formas funcionan.

```
app.get('/', (req, res) => {
  res.sendFile(__dirname + '/public/index.html');
});
```

```
app.get('/', (req, res) => {
  res.sendFile(`${__dirname}/public/index.html`);
});
```